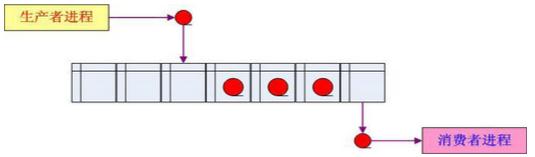


计算机与信息学院实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验课程： | 操作系统实验 | | | | |
| 实验编号： |  | | | | |
| 实验名称： | 生产者与消费者 | | | | |
| 实验人员： | 年级 | | 2017 | | |
| 专业 | | 软件工程 | | |
| 学号 | | 17111205043 | | |
| 姓名 | | 李志威 | | |
| 实验日期： | 2019.4.1 | | | | |
| 上交日期： | 2019.4.8 | | | | |
| 实 验 室： | 2030201 | | | | |
| 实验评价： |  | | | | |
|  | 实验成绩： |  | | 评定日期： |  |
|  | 指导教师： |  | | | |

一、实验目的



通过实验，掌握Windows环境下互斥锁和信号量的实现方法，进一步加深对进程同步机制的理解。

二、实验环境

（1）查资料学习如：创建线程的函数\_beginthread( )、创建信号量函数CreateSemaphore( )等，了解这几个函数的使用方法。

（2）熟悉并理解生产者/消费者问题。

Devc++

三、实验内容与步骤

（1）问题描述

编程实现生产者/消费者的同步问题。

（2）基本要求

 利用线程模拟进程；

 可视化显示模拟的同步。

 实验报告中需分析所得结果，指出某消费者线程或生产者线程在何时被阻塞、何时被唤醒

提示：

（1）设计在同一个进程地址空间内执行的多个线程：生产者线程和消费者线程；

（2）生产者线程生产物品，然后将物品放置在一个空缓冲区中供消费者线程消费；

（3）消费者线程从缓冲区中获得物品，然后释放缓冲区；

（4）生产者线程生产物品时，若无空缓冲区可用，生产者线程必须等待消费者线程释放出一个空缓冲区；

（5）消费者线程消费物品时，若无物品，消费者线程将被阻塞，直到新的物品被生产出来。

（6）需要使用如下三个信号量：

 一个互斥信号量，用以阻止生产者线程和消费者线程同时操作缓冲区列表；

 一个信号量，用于当缓冲区满时让生产者线程等待；

 一个信号量，用于当缓冲区空时让消费者线程等待。

（7）学有余力的同学可同时实现在Linux操作系统下的生产者消费者问题。

（8）\_beginthread函数说明

\_beginthread( void( \*start\_address )( void \* ), unsigned stack\_size, void \*arglist) //创建线程

 start\_address：程序执行一个新线程的起始地址

 stack\_size：新线程的堆栈大小或0

 Arglist：传给新线程的变量清单或空

（9）信号量包含的几个操作原语：

 CreateSemaphore(SecAttr, InitialCoutn, MaxCount, SemName) //创建一个信号量

 SecAttr：安全属性指针

 InitialCoutn：信号量初值

 MaxCount：允许的最大值

 SemName：信号量对象名指针，若SemName参数为NULL时，信号量局限于一个进程中；若给予一个字符串对象名，则其它进程也可以使用该信号量。

 例如：mutex = CreateSemaphore(NULL,1,1,NULL)，则mutex就是信号量

 ReleaseSemaphore(Semaphore, ReleaseCount, PreviousCount ) //释放信号量，就是V操作

 Semaphore：信号量

 ReleaseCount：递增数量。若为1，则每次增加1

 PreviousCount：先前的值，若不需要信号量上次的值，则这个参数可置为NULL

 WaitForSingleObject(hHandle, dwMilliseconds) //等待信号，就是P操作

 hHandle：对象句柄，可以指定一系列的对象，如Event、Job、Memory resource notification、Mutex、Process、Semaphore、Thread、Waitable timer等，在本实验中就是信号量。

 dwMilliseconds：定时时间间隔，单位为毫秒。若指定一个非零值，函数处于等待状态直到hHandle标记的对象被触发，或者时间到了。若为0，对象没有被触发信号，函数不会进入一个等待状态，它总是立即返回。若为INFINITE，对象被触发信号后，函数才会返回。

四、实验结果与分析

#include <stdio.h>

#include <process.h>//\_beginthread函数

#include <windows.h>

#include <time.h>

//srand(time(NULL));

//x=rand()%6;

using namespace std;

#define N 5

//可视化设置队列

int buff[N];

/\* 有N个槽数的缓冲区buf[N]，并实现循环缓冲队列 \*/

int front=0, rear=0;

/\*

ReleaseSemaphore(mutex,1,NULL);//v操作

WaitForSingleObject(mutex,INFINITE);//p操作

注意：在使用WaitForSingleObject等待信号量句柄时，

若信号量为signal状态，则wait过后信号量自动减1，

直到使用ReleaseSemaphore释放信号量，信号量才可增加

\*/

int x;

int p\_num=0;

int c\_num=0;

HANDLE mutex=CreateSemaphore(NULL,1,1,NULL);

//信号量初值1，最大值1

HANDLE full=CreateSemaphore(NULL,0,5,NULL);

HANDLE empty=CreateSemaphore(NULL,5,5,NULL);

void produce\_item()

//生产产品

{

printf("生产第%d一个产品 \n",++p\_num);

}

void enter\_item()

//放入缓存区

{

printf("放入缓存区第[%d]\n",rear);

buff[rear]=1;

rear=(rear+1)%N;

if(rear==front)

printf("缓存区已满\n");

}

void remove\_item()

//从缓存区中取

{

printf(" 从缓存区中取[%d] \n",front);

buff[front]=1;

front=(front+1)%N;

}

void consume\_item()

//消费产品

{

printf(" 消费产品第%d个商品\n",++c\_num);

}

/\* 生产者 \*/

void producer(PVOID param)

{

while(1){

Sleep(1000);

// x=rand()%7+1;printf("\n%d\n",x);

// Sleep(x\*1000);

produce\_item();

WaitForSingleObject(empty,INFINITE);//p操作

WaitForSingleObject(mutex,INFINITE);//p操作

enter\_item(); /\* 将一个新的数据项放入缓冲区 \*/

ReleaseSemaphore(mutex,1,NULL);//v操作

ReleaseSemaphore(full,1,NULL);//v操作

}

}

/\* 消费者 \*/

void consumer(PVOID param)

{

while(1){

//Sleep(8000);

x=rand()%6+1;printf("\n%d\n",x);

Sleep(x\*1000);

WaitForSingleObject(full,INFINITE);//p操作

WaitForSingleObject(mutex,INFINITE);//p操作

remove\_item(); /\* 从缓冲区中取走一个数据项 \*/

ReleaseSemaphore(mutex,1,NULL);//v操作

ReleaseSemaphore(empty,1,NULL);//v操作

consume\_item(); /\* 对数据项进行操作（消费）\*/

}/\* 对数据项进行操作（消费）\*/

}

int main(){

srand((unsigned)time(0));

\_beginthread(producer, 0, NULL);

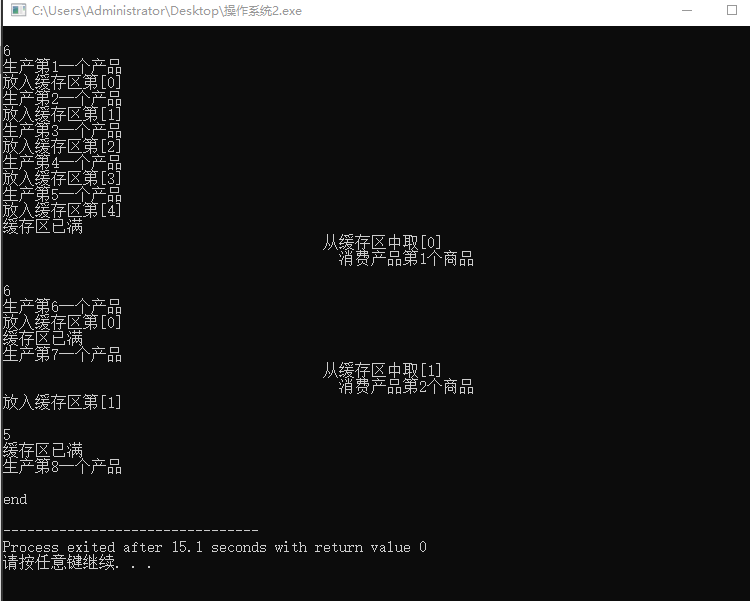
\_beginthread(consumer, 0, NULL);

Sleep(15000);

printf("\nend\n") ;

return 0;

}



一共休眠15秒，生产5个花费5秒（空闲一秒）-6

取一个1秒-1

生产一个（空闲5秒）-6

取一个1秒-1

生产一个-1